

This correspondence is being deposited with the United States Postal Service as Express Mail addressed to: Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on Sept. 25, 2003, Express Mail Receipt No. ER264470395 US.

1       MULTI-SERVICE SEGMENTATION AND REASSEMBLY DEVICE HAVING  
2       INTEGRATED SCHEDULER AND ADVANCED MULTI-TIMING WHEEL SHAPER

4                               Rami Zecharia

5                               Bidyut Parruck

6                               Chulanur Ramakrishnan

7       CROSS-REFERENCE TO RELATED APPLICATION

8       This application claims the benefit under 35 U.S.C. §119 of Provisional  
9       Application 60/434,554, filed December 18, 2002. The entire content of Provisional  
10      Application 60/434,554 is incorporated herein by reference.

12      DETAILED DESCRIPTION

13      **Figure 1** is a simplified diagram of a router 100 in accordance with an  
14      embodiment of the present invention. Router 100 includes a plurality of line cards  
15      101-104, a switch fabric 105 and a central processing unit (CPU) 106. The line  
16      cards 101-104 are coupled to switch fabric 105 by buses 107-114. CPU 106 is  
17      coupled to line cards 101-104 by another parallel bus 115. In the present example,  
18      parallel bus 115 is a 32-bit PCI bus. In this example, each of the line cards can  
19      receive network communications in multiple formats. For example, line card 101 is  
20      coupled to a fiber optic cable 116 such that line card 101 can receive from cable 116  
21      network communications at OC-192 rates in packets and/or ATM cells.

22      Line card 101 is also coupled to a fiber optic cable 117 such that line card 101  
23      can output onto cable 117 network communications at OC-192 rates in packets  
24      and/or ATM cells. All the line cards 101-104 in this example have substantially  
25      identical circuitry.

26      **Figure 2** is a more detailed diagram of representative line card 101. Line card  
27      101 includes OC-192 optical transceiver modules 118 and 119, two serial-to-parallel  
28      devices (SERDES) 120 and 121, a framer integrated circuit 122, an IP classification  
29      engine 123, two multi-service segmentation and reassembly devices (MS-SAR

1 devices) 124 and 125, static random access memories (SRAMs) 126 and 127,  
2 dynamic random access memories (DRAMs) 128 and 129, and a switch fabric  
3 interface 130. IP classification engine 123 may, in one embodiment, be a  
4 classification engine available from Fast-Chip Incorporated, 950 Kifer Road,  
5 Sunnyvale, CA 94086. Framer 122 may, in one embodiment, be a Ganges S19202  
6 STS-192 POS/ATM SONET/SDH Mapper available from Applied Micro Circuits  
7 Corporation, 200 Brickstone Square, Andover, MA 01810. MS-SAR devices 124  
8 and 125 are identical integrated circuit devices, one of which (MS-SAR 124) is  
9 configured to be in an "ingress mode", the other of which (MS-SAR 125) is  
10 configured to be in an "egress mode". Each MS-SAR device includes a mode  
11 register that is written to by CPU 106 via bus 115. When router 100 is configured,  
12 CPU 106 writes to the mode register in each of the MS-SAR devices on each of the  
13 line cards so as to configure the MS-SAR devices of the line cards appropriately.

14 Fiber optic cable 116 of Figure 2 can carry information modulated onto one or  
15 more of many different wavelengths (sometimes called "colors"). Each wavelength  
16 can be thought of as constituting a different communication channel for the flow of  
17 information. Accordingly, optics module 118 converts optical signals modulated onto  
18 one of these wavelengths into analog electrical signals. Optics module 118 outputs  
19 the analog electrical signals in serial fashion to Serdes 120. Serdes 120 receives  
20 this serial information and outputs it in parallel form to framer 122. Framer 122  
21 receives the information, frames it, and outputs it to classification engine 123 via  
22 SPI-4 bus 131. Classification engine 123 performs IP classification and outputs the  
23 information to the ingress MS-SAR 124 via another SPI-4 bus 132. The ingress MS-  
24 SAR 124 processes the network information in various novel ways (explained  
25 below), and outputs the network information via to switch fabric 105 (see Fig. 1) via  
26 SPI-4 bus 133, switch fabric interface 130, and bus 107. All the SPI-4 buses of  
27 Figures 1 and 2 are separate SPI-4, phase II, 400 MHz DDR buses having sixteen  
28 bit wide data buses.

29 Switch fabric 105, once it receives the network information, supplies that  
30 information to one of the line cards of router 100. Each of the line cards is identified  
31 by a "virtual output port" number. To facilitate the rapid forwarding of such network  
32 information through the switch fabric 105, network information passed to the switch

1 fabric 105 for routing is provided with a "switch header". The "switch header" may  
2 be in a format specific to the manufacturer of the switch fabric of the router. The  
3 switch header identifies the "virtual output port" to which the associated network  
4 information should be routed. Switch fabric 105 uses the virtual output port number  
5 in the switch header to route the network information to the correct line card.

6 Router 100 determines to which of the multiple line cards particular network  
7 information will be routed. Accordingly, the router's CPU 106 provisions lookup  
8 information in (or accessible to) the ingress MS-SAR 124 so that the MS-SAR 124  
9 will append an appropriate switch header onto the network information before the  
10 network information is sent to the switch fabric 105 for routing. Switch fabric 105  
11 receives the network information and forwards it to the line card identified by the  
12 particular "virtual output port" in the switch header. The network information and  
13 switch header is received onto the egress MS-SAR of the line card that is identified  
14 by the virtual output port number in the switch header.

15 For explanation purposes, MS-SAR 125 in Figure 2 will represent this egress MS-  
16 SAR. The egress MS-SAR 125 receives the network information, removes the  
17 switch header, performs other novel processing (explained below) on the network  
18 information, and outputs the network information to framer 122. Framer 122 outputs  
19 the network information to serdes 121. Serdes 121 converts the network information  
20 into serial analog form and outputs it to output optics module 119. Output optics  
21 module 119 converts the information into optical signals modulated onto one  
22 wavelength channel. This optical information is then transmitted from router 100 via  
23 fiber optic cable 117.

#### 24 25 MS-SAR IN MORE DETAIL:

26 **Figure 3** is a more detailed diagram of an MS-SAR device 124 in accordance  
27 with an embodiment of the present invention. MS-SAR device 124 includes an  
28 incoming interface block 201, a lookup engine block 202, a segmentation block 203,  
29 a memory manager block 204, a reassembly and header-adding block 205, an  
30 outgoing interface block 206, a per flow queue (PFQ) block 207, a class-based  
31 weighted fair queuing (CBWFQ) block 208, a data base (DBS) block 209, a traffic  
32 shaper block 210, an output scheduler block 211, and a CPU interface block 212.

1 MS-SAR 124 interfaces to and uses numerous other external memory integrated  
2 circuit devices 213-220 that are disposed on the line card along with the MS-SAR.

3 In operation, MS-SAR 124 receives a flow of network information via input  
4 terminals 221. When incoming interface block 201 accumulates a sufficient amount  
5 of the network information, it forwards the information to lookup block 202. CPU 106  
6 (see Fig. 1) has previously placed lookup information into MS-SAR 124 so that  
7 header information in the incoming network information (in the case of MS-SAR  
8 being used in the ingress mode) can be used by lookup block 202 to find: 1) a  
9 particular flow ID (FID) for the flow that was specified by CPU 106, and 2) an  
10 application type. The application type, once determined, is used by other blocks of  
11 MS-SAR 124 to configure themselves in the appropriate fashion to process the  
12 network information appropriately.

13 The FID and application type, once determined, are passed to segmentation  
14 block 203. Segmentation block 203 performs various operations on the associated  
15 network information and then forwards the information to memory manager block  
16 204.

17 External payload memory 213 contains a large number of 64-byte buffers, each  
18 buffer being addressed by a buffer identifier (BID). When memory manager block  
19 204 receives a 64-byte chunk (also called a "cell") of information associated with the  
20 flow, memory manager block 204 issues an "enqueue" command via enqueue  
21 command line 222 to per flow queue block 207. This constitutes a request for the  
22 per flow queue block 207 to return the BID of a free buffer. Per flow queue block  
23 207 responds by sending memory manager block 204 the BID of a free buffer via  
24 lines 223. Memory manager block 204 then stores the 64-byte chunk of information  
25 in the buffer in payload memory 213 identified by the BID.

26 Per flow queue block 207 maintains a linked list (i.e., a "queue") of the BIDs for  
27 the various 64-byte chunks of each flow that are stored in payload memory 213.  
28 Such a linked list is called a "per flow queue". Once the linked list (queue) for the  
29 flow is formed, the linked list can be popped (i.e., dequeued) in a particular way and  
30 at such a rate that the associated chunks of information stored in payload memory  
31 213 are output from MS-SAR 124 in a desired fashion. To perform a dequeue  
32 operation, per flow queue block 207 accesses the per flow queue of the flow ID,

1 determines the next BID for the FID to be dequeued, and outputs that BID in the  
2 form of a “dequeue command” to memory manager block 204. Memory manager  
3 block 204 uses the BID to retrieve the identified chunk from payload memory 213  
4 and outputs that chunk to reassembly block 205. Reassembly block 205 performs  
5 other actions on the chunk and then outputs the chunk from MS-SAR 124 via  
6 outgoing interface block 206 and output terminals 224.

7 It is therefore seen that the output from MS-SAR 124 of chunks (i.e., cells) for a  
8 particular FID can be controlled by controlling when dequeue commands for the FID  
9 are sent to memory manager block 204. Operation of the remaining blocks (207-  
10 211) of MS-SAR 124 is directed to a “control path” whereby this dequeuing process  
11 is controlled so as to achieve desired traffic shaping, traffic scheduling, traffic  
12 policing, and traffic metering functions.

#### 13 14 SIMPLIFIED OVERVIEW OF CONTROL PATH INPUT PHASE OPERATION:

15 Operation of the control path portion of MS-SAR 124 is explained in terms of an  
16 “input phase” and an “output phase”. Before a chunk for an FID is received and  
17 stored in payload memory 213, MS-SAR 124 is first provisioned with information on  
18 how the FID is to be shaped and/or scheduled. This provisioning is done via CPU  
19 interface block 212.

20 An input phase begins when a chunk for an FID (FID3 in this example) is to  
21 be stored in payload memory 213. Per flow queue (PFQ) block 207 supplies a BID  
22 to memory manager block 204 and then links the BID to the per flow queue for the  
23 particular FID. PFQ block 207 then forwards the FID to CBWFQ block 208 via lines  
24 235. We assume now for ease of explanation in this simplified introductory example  
25 that CBWFQ block 208 does not merge the FID with any other FID. The FID  
26 therefore passes through CBWFQ block 208 to DBS block 209 via lines 236. MS-  
27 SAR 124 in this example has been provisioned beforehand to shape FID3 (rather  
28 than to schedule FID3). DBS block 209 includes a DBS internal FID memory 225  
29 that is provisioned beforehand to contain, for each FID, a set of parameters.

30 **Figure 4** is a diagram of one such set of parameters in DBS internal FID  
31 memory 225. One parameter is a Rate\_ID. The Rate\_ID value stored for the FID  
32 identifies one of a set of rate variables. Each of these sets of rate variables is called

1 a "rate profile". The rate profiles are stored in shaper internal Rate\_ID memory 226.  
2 Each profile is identified by its own Rate\_ID.

3 **Figure 5** is a diagram of one rate profile (for one Rate\_ID) as the profile is  
4 stored in shaper internal Rate\_ID memory 226. The various rate variables of the  
5 profile determine how shaper portion 227 of shaper block 210 will shape the  
6 associated FID. Using the FID number (FID3 in this case) as the base address,  
7 DBS block 209 looks up the Rate\_ID value stored in DBS internal FID memory 225  
8 for FID3, and then forwards that Rate\_ID along with the FID number and other FID-  
9 specific values to both shaper block 210 as well as to scheduler block 211. The  
10 information is sent to shaper block 210 via lines 237. The information is sent to  
11 scheduler block 211 via lines 238. Two additional bits are also sent to indicate that  
12 the shaper block, and not the scheduler block, is to perform an input phase for FID3.

13 Shaper block 210 shapes the incoming FID3 with a particular rate identified  
14 by the Rate\_ID value by first linking FID3 in a "shaper input phase" to an  
15 appropriately distant future "slot" on a "timing wheel". **Figure 6** is a conceptual  
16 diagram of a timing wheel 300 before FID3 is linked to it. A different linked list of  
17 FIDs can be linked to each of the various slots of timing wheel 300. Conceptually,  
18 the timing wheel rotates at a constant rate such that the slot number for each slot is  
19 decremented once each slot time. In this example, a slot time is eight cycles of the  
20 200 MHz system clock. When the slot to which an FID is linked becomes slot zero,  
21 then all FIDs linked to that slot are output from the wheel. Accordingly, the future  
22 slot to which the incoming FID3 is linked in this example will determine the amount  
23 of delay until FID3 will be output. If FID3 is linked to a slot well into the future, then it  
24 will take longer for the wheel to rotate to that slot. The particular slot to which FID3  
25 is linked therefore determines the rate at which FID3 will be shaped. The shaper  
26 input phase involves calculating the particular future slot to which FID3 will be linked  
27 in order to achieve the programmed shaping rate determined by the Rate\_ID.

28 Using the rate information retrieved from internal Rate\_ID memory 226 as  
29 well as other information for the FID stored in shaper internal FID#1 and FID#2  
30 memories 228 and 229, traffic shaper portion 227 determines the future time slot to  
31 which FID3 should be linked. **Figure 7** is a diagram of shaper internal FID#1  
32 memory 228. **Figure 8** is a diagram of shaper internal FID#2 memory 229.

**Figure 9** is a diagram illustrating how shaper block 210 links FID3 to wheel 300. In the present example, shaper portion 227 determines that FID3 is to be linked to slot number six. There is already a linked list of two FIDs (FID1 and FID2) linked to slot number six. As illustrated, for each slot on the wheel there is a SLOT\_RP read pointer and a SLOT\_WP write pointer. The slot read and slot write pointers for slot six point to the associated linked list of FIDs. The read and write slot pointers for all the slots of the wheel are stored in shaper external slot memory 215. **Figure 10** is a diagram of the pair of read and write slot pointers for one slot on one wheel as that pair of slot pointers is stored in shaper external slot memory 215.

To add FID3 to the linked list on slot number six, the SLOT\_WP write pointer is changed to point to FID3. This is indicated in Figure 9 by dashed line 304. Each FID linked to a slot has a FID\_NEXT pointer that can be set to point to a subsequent FID in a linked list. The FID\_NEXT pointer for each FID is stored in shaper internal FID#2 memory 229 (see Figure 8). To complete the linking of FID3 to the linked list on slot number six, the FID\_NEXT pointer for FID2 is changed to point to FID3. This is indicated in Figure 9 by dashed line 305. With the slot write pointer SLOT\_WP set to point to added FID3 and with the FID\_NEXT pointer for FID2 set to point to the added FID3, FID3 is linked to slot number six as illustrated in Figure 9.

As set forth above, timing wheel 300 rotates at a constant rate of one slot time per every eight cycles of the 200 MHz system clock. When the slot at which FID3 is linked reaches the zero position, then FID3 is output from wheel 300 and is pushed into a "shaper output FIFO" in shaper portion 227. In this way, the timing wheel 300 continues to rotate and to fill the wheel's shaper output FIFO.

**Figure 11** is a diagram of eight timing wheels implemented by shaper block 210. Wheel 1 is the highest priority wheel, wheel 2 is the next highest priority wheel, and so forth. The eight timing wheels all rotate in unison at a constant rate. As illustrated, each of the eight timing wheels has its own "shaper output FIFO" into which it places FIDs. Shaper output FIFO 301 is the shaper output FIFO for the eighth timing wheel 300.

MS-SAR 124 is provisioned such that each FID to be shaped is preprogrammed to go out on an assigned output port. The output port number for each FID is stored in DBS internal FID memory 225. The output port number for

1 FID3 was previously passed by DBS block 209 over lines 237 to shaper block 210  
2 along with the FID. One by one, shaper portion 227 moves FIDs from the "shaper  
3 output FIFOs" to an associated plurality of "per-port output FIFOs" 303 in DBS block  
4 209. Provided an FID is present in a shaper output FIFO, there is one such FID  
5 moved per wheel during each slot time. As illustrated in Figure 11, there are sixty-  
6 four such "per-port output FIFOs" in DBS block 209 for each wheel, there being one  
7 "per-port output FIFO" for each of the sixty-four possible output ports. The per-port  
8 output FIFOs 303 in DBS block 209 therefore form an 8x64 matrix of per-port output  
9 FIFOs. The particular per-port output FIFO to which the FID is moved is determined  
10 by the output port number stored for FID3 in FID memory 225.

11 Figure 11 illustrates how this is done. For each FID stored in a per-port  
12 output FIFO, an associated "DBS credit" value is also stored. If the FID to be  
13 moved into a per-port output FIFO is already present in the per-port output FIFO,  
14 then the associated "DBS credit" number for that FID is incremented. The "DBS  
15 credit" for the FID therefore accumulates at the configured shaping rate.

16 When an FID is moved from a shaper output FIFO to a per-port output FIFO,  
17 the FID can either be "not-empty" (DBS block 209 indicates that there are more cells  
18 for this FID) or the FID can be "empty" (DBS block 209 indicates that there are no  
19 more cells for this FID). If the FID is "not-empty" then the FID is reattached to the  
20 timing wheel at a new time slot. The new slot is calculated based on the Rate\_ID for  
21 the FID, how many slot times the FID was sitting in the shaper output FIFO waiting  
22 to be moved to a per-port output FIFO, and some other parameters. If the FID is  
23 "empty", then the FID is not reattached. In this way, the FIDs of the chunks (cells)  
24 being stored in payload memory 213 are placed by shaper block 210 into the per-  
25 port output FIFOs in DBS block 209.

26 In the simplified example described so far, MS-SAR 124 was provisioned to  
27 shape FID3. If rather than shaping FID3, MS-SAR 124 had been provisioned to  
28 schedule FID3, then the input phase may have proceeded in accordance with the  
29 simplified input phase set forth below. As in the example above, DBS block 209  
30 initially forwards the FID (FID3 in this case) to both shaper block 210 as well as  
31 scheduler block 211. In this example, however, the two additional bits that



1 accompany the FID would indicate that the scheduler, and not the shaper, is to  
2 perform an input phase for FID3.

3 Upon receiving the FID, scheduler block 211 links the FID into a linked list of  
4 FIDs maintained for a single priority class and a single output port. The priority class  
5 is called a "quality of service" (QOS). There are eight possible QOSs. Accordingly,  
6 for each port, there can be up to eight such linked lists of FIDs (one linked list for  
7 each QOS).

8 For each FID, a QOS\_ADDRESS is provisioned beforehand into scheduler  
9 external FID memory 216. This QOS\_ADDRESS contains three bits that identify the  
10 one QOS assigned to this FID, and eight bits that identify the output port to which  
11 this FID is to be scheduled. **Figure 12** is a diagram of the fields in scheduler  
12 external FID memory 216 that pertain to one FID.

13 The QOS\_ADDRESS also points to one of a plurality of "QOS descriptors" in  
14 an internal QOS parameter/descriptor memory 232. **Figure 13** is a diagram of the  
15 QOS descriptor portion of the scheduler internal QOS par/descriptor memory 232  
16 and **Figure 14** is a diagram of the QOS parameter portion of the scheduler internal  
17 QOS par/descriptor memory 232. The QOS descriptor pointed to by  
18 QOS\_ADDRESS identifies a read pointer F\_RP that points to the head of the linked  
19 list of FIDs for the QOS and a write pointer F\_WP that points to the tail of the linked  
20 list of FIDs for the QOS. Scheduler block 211 uses these pointers to link the  
21 incoming FID3 into the correct linked list of FIDs (the linked list for the indicated  
22 QOS and for the correct output port). Scheduler block 211 does this by updating the  
23 read and write pointers for the QOS (stored in QOS par/descriptor memory 232) in a  
24 fashion analogous to how the FID was added to the linked list connected to slot six  
25 of timing wheel 300 as described above.

26 In addition to linking the incoming FID3 into the correct linked list of FIDs, the  
27 scheduler block 211 also sets a bit associated with the correct output port to indicate  
28 that the correct output port now has traffic (i.e., is now not empty). Scheduler block  
29 211 does this by writing an appropriate value into an eight-bit QW\_EMPTY field in  
30 an internal port parameter/descriptor memory 233. There is one bit in the  
31 QW\_EMPTY field for each QOS of the output port. **Figure 15** is a diagram of the  
32 scheduler internal port parameter memory portion of the port par/descriptor memory

233, and **Figure 16** is a diagram of the scheduler internal port descriptor memory portion of the port par/descriptor memory 233. Once the QW\_EMPTY field is been updated, the input phase is concluded. This concludes the simplified overview of the input phase of the control path.

#### SIMPLIFIED OVERVIEW OF CONTROL PATH OUTPUT PHASE OPERATION:

**Figure 17** is a diagram that illustrates a port calendar 230 that is located in DBS block 209. An output phase begins when this port calendar 230 informs shaper block 210 and scheduler block 211 of an output port that is due for dequeue processing. Port calendar 230 can be conceptualized as a rotating list where each row entry indicates an output port. There can be up to 96 row entries in the list. The row entries in port calendar 230 are serviced one by one down the list until a row entry is encountered that has its "jump" bit set. The jump bit being set causes the next row entry serviced to be the first row entry in the calendar. The servicing of row entries is therefore done in a round robin fashion. Each row entry corresponds to the bandwidth capacity of STS-1. Each row entry is serviced in eight clocks of the 200 MHz system clock. If it is desired to dedicate a greater percentage of bandwidth to one output port than to other output ports, then the one output port may be designated in more than one row in port calendar 230. For example, to configure various of the MS-SAR output ports to have STS-1, STS-3, and STS-12 bandwidths, the STS-1 output ports would be assigned one row each in the port calendar, the STS-3 output ports would be assigned three rows each in the port calendar, and the STS-12 output ports would be assigned twelve rows each in the port calendar. In the example set forth in Figure 17, port calendar 230 holds one row entry for Port 0 (an STS-1 port) but it holds three row entries for Port 1 (an STS-3 port).

Once port calendar 230 has identified an output port for servicing, the output port number is sent to the shaper block 210 and to the scheduler block 211. Either the shaper block 210 or the scheduler block 211, or both, may then undergo output phases to provide FIDs back to DBS block 209 for dequeuing. If both the shaper block 210 and the scheduler block 211 provide FIDs, then DBS block 209 accepts the FID provided by shaper block 210 for dequeuing. If DBS block 209 accepts the FID from shaper block 210 when scheduler block 211 has also provided an FID,

1 then the output phase of scheduler block 211 is aborted such that scheduler block  
2 211 cannot change any values in memories 232, 233 or 216. By not allowing the  
3 values in memories 232, 233 and 216 to change, the output phase of scheduler  
4 block 211 is effectively reversed as if it never happened.

5 Output phase operation of shaper block 210 is now explained in more detail in  
6 connection with Figure 11. As described previously, shaper block 210 in the input  
7 phase placed FIDs into the 8x64 matrix of per-port output FIFOs 303 located in DBS  
8 block 209. Now, in the output phase, FIDs are removed one by one from the per-  
9 port output FIFOs 303 in strict priority fashion. For example, an FID will be removed  
10 from a per-port output FIFO of the highest priority wheel (wheel one) if there is an  
11 FID in the associated per-port output FIFO for the selected port. If there are no FIDs  
12 in the per-port output FIFO for the selected port for wheel one (the highest priority  
13 wheel), then an FID is removed from the per-port output FIFO of wheel two for the  
14 selected port provided there is an FID in that per-port output FIFO. If there are no  
15 FIDs in the per-port output FIFOs for either wheel one or for wheel two for the  
16 selected port, then an FID can be removed from the per-port output FIFO of wheel  
17 three for the selected port, and so forth.

18 When DBS block 209 removes a FID from a per-port output FIFO, the DBS  
19 block 209 decrements the associated "DBS credit" value. As set forth above in the  
20 explanation of the input phase, the "DBS credit" value is incremented in the input  
21 phase at the configured shaping rate of the FID. The "DBS credit" value therefore  
22 indicates whether the shaper is lagging behind the unloading of the per-port output  
23 FIFOs or whether the shaper is leading the unloading of the per-port output FIFOs.  
24 If the shaper is lagging behind to a sufficient degree, then the "DBS credit" value  
25 may reach a negative value. If an EOP for such a shaped FID is reached and the  
26 associated "DBS credit" value is negative, then DBS block 209 does not continue  
27 sending this FID out (unloading this FID from the per-port output FIFO in subsequent  
28 output phases). Rather, DBS 209 suspends the unloading of this FID again until the  
29 shaper has incremented the DBS credit for this FID back up to a positive value.

30 Cells of different packets cannot be interleaved as they are output from an  
31 output port. Accordingly, once DBS block 209 has started removing an FID from a  
32 per-port output FIFO (whichever it picked from priority), it will not switch to start

1 removing another FID within the same output port until it receives an EOP indication  
2 (indicating the last cell of the packet) back from PFQ block 207. DBS block 209 will  
3 also not switch from unloading a per-port output FIFO from one priority wheel to  
4 unloading a per-port output FIFO from another priority wheel until the EOP indication  
5 is reached. DBS block 209 is informed of the EOP indication via PFQ block 207 and  
6 line 234. If an EOP indication is not received for the current output phase, then DBS  
7 block 209 just decrements the "DBS credit" value associated with the FID and sends  
8 the FID to PFQ block 207 via CBWFQ block 208.

9 If, on the other hand, DBS block 209 receives an EOP for the current output  
10 phase, then there are two possibilities. If an EOP indication is received and the  
11 "DBS credit" is negative, then the FID is removed from the per-port output FIFO.  
12 The DBS credit being negative indicates that the shaper wheel is running slower  
13 than the unloading of per-port output FIFOs by DBS block 209. The FID is therefore  
14 not dequeued again until the negative DBS credit is incremented back to positive  
15 one. If an EOP indication is received and the "credit" is positive, then the "DBS  
16 credit" value is decremented and the FID is left in the per-port output FIFO. In this  
17 way, DBS block 209 removes FIDs from the per-port output FIFOs 303, decrements  
18 the associated "DBS credit" values, and forwards the FIDs to CBWFQ block 208 via  
19 lines 239.

20 For ease of explanation, we assume in this example that CBWFQ block 208  
21 has not performed any merging of FIDs. The FID therefore passes through CBWFQ  
22 block 208 unchanged and is supplied to PFQ block 207 via lines 240. PFQ block  
23 207 receives the FID, performs a "dequeue" operation on the queue for the indicated  
24 FID, and retrieves the BID of the next cell. The BID is then forwarded to memory  
25 manager block 204 in the form of a "dequeue command" via lines 223. PFQ  
26 maintains the per flow queues and a free buffer queue in external memories 218-  
27 220. Memory manager block 204, upon receiving the "dequeue command" for the  
28 BID, retrieves from payload memory 213 the cell data from the buffer identified by  
29 the BID. The retrieved cell data is then sent out of MS-SAR 124 via reassembly and  
30 header adding block 205 and outgoing interface block 206.

31 If shaper block 210 does not supply a FID back to DBS block 209 for the  
32 output port identified by port calendar 230, then a FID may be supplied by an output

1 phase of scheduler block 211. Having an FID “scheduled” means that the flow will  
2 attempt to use all the free bandwidth available. The performance of a scheduled FID  
3 depends on the available bandwidth and the FID’s own characteristics with respect  
4 to the other active flows in the system. As described above in connection with the  
5 input phase, every FID in the system is assigned a QOS class (the QOS class  
6 determines the relative priority of the FID with respect to other FIDS in other QOS  
7 classes) and an output port. Each output port may have an associated plurality of  
8 non-empty QOSs, and each such associated non-empty QOS may have a linked list  
9 of FIDs. The function of the scheduler is to choose one of the non-empty QOS  
10 classes for the output port, and then to choose one of the FIDs belonging to that  
11 QOS class. The resulting FID is the FID returned to DBS block 209.

12 Every output port in the system can be provisioned to have its own scheduling  
13 algorithm to choose the QOS class. The allowed scheduling algorithms are 1) strict  
14 priority, 2) weighted round robin, or 3) a mixture of both. For each output port, one  
15 QOS (the QOS number seven) is neither a strict priority QOS nor a weighted round  
16 robin QOS, but rather is reserved as a “best effort” QOS. The mixture of algorithms  
17 is provisioned by setting several of the highest seven priority QOS classes of a port  
18 to be selected between using the strict priority scheme, and setting the lower ones of  
19 the seven priority QOS classes of the port to be selected between using the  
20 weighted round robin scheme.

21 To select the QOS for the output port designated by port calendar 230, the  
22 scheduler block 211 uses the output port number to read a PREV\_QOS field in the  
23 port par/descriptor memory 233 (see Figure 16). This PREV\_QOS field stores a  
24 three-bit value that designates the QOS that was services last for the output port.  
25 Once the scheduling out of FIDs for a QOS has started, the QOS number cannot be  
26 changed until an EOP indication has been received back from PFQ block 207.  
27 Accordingly, if no EOP is received back from PFQ block 207 for this output phase,  
28 then the QOS selected by output scheduler 211 is the previous QOS designated by  
29 PREV\_QOS. If, on the other hand, an EOP for this QOS has been received, then a  
30 different QOS can be chosen as determined by the predetermined algorithm.

31 For each output port, the scheduler port parameter memory portion of the port  
32 par/descriptor memory 233 (see Figure 15) stores an eight-bit PRIORITY field.

1 There is one bit in this field for each of the eight QOSs of the port. Setting the bit  
2 associated with a QOS to a "1" designates the QOS as a strict priority QOS. Setting  
3 the bit associated with a QOS to a "0" designates the QOS as a weighted round  
4 robin QOS. The output scheduler block 211 uses the output port number received  
5 from port calendar 230 to look up the eight-bit PRIORITY field for the designated  
6 output port.

7 A QOS will be selected from the QOSs designated as strict priority QOSs if  
8 one of those QOSs is designated as being "not empty". The output scheduler  
9 determines whether a QOS is empty by reading the bits in the QA\_EMPTY field (see  
10 Figure 16) in the port par/descriptor memory 233.

11 If a strict priority QOS is not selected, then output scheduler block 211  
12 attempts to select a QOS from the QOSs designated as weighted round robin QOSs  
13 by the eight-bit PRIORITY field for the output port. To implement the weighted  
14 round robin scheme, a queue of QOSs is maintained for the output port. The three-  
15 bit value ACTIVE\_PTR stored in port par/descriptor memory 233 identifies the next  
16 QOS in the queue to be serviced. If there is no QOS to select, then the best efforts  
17 QOS seven is selected to be the QOS.

18 Once a QOS is chosen, then output scheduler block 211 chooses one of the  
19 FIDs in the linked list of FIDs linked to the chosen QOS of the selected output port.  
20 To find the FID, the port number is multiplied by the number eight and the QOS  
21 number is added to this product. The result is an address that points to the F\_RP  
22 read pointer (see Figure 13) in the QOS par/descriptor memory 232. This F\_RP  
23 read pointer points to the head of the linked list of FIDs that is linked to the selected  
24 QOS of the selected output port. Output scheduler 211 outputs this FID to DBS  
25 block 209 as the selected FID.

26 Once the FID is chosen, scheduler block 211 forwards the FID to DBS block  
27 209. DBS block 209 determines whether the FID from the scheduler or a FID from  
28 the shaper will be sent out. If there is a FID from the shaper, then the FID from the  
29 shaper is sent out and the DBS causes the output phase of the scheduler to abort,  
30 thereby preventing the scheduler from updating any parameters and essentially  
31 undoing the scheduler output phase. If, on the other hand, there is no FID from the

1 shaper, then the FID from the scheduler is sent out and the scheduler is allowed to  
2 update its parameters.

3

#### 4 DATA BASE BLOCK IN MORE DETAIL:

5 MS-SAR 124 is provisioned such that port calendar 230 operates in one of  
6 two selectable modes: a non-work conserving mode, and a work-conserving mode.  
7 **Figure 18** is a diagram of a port calendar memory located in DBS block 209 that is  
8 used to implement port calendar 230.

9 Every sixteen 200 MHz system clocks, there can be one FID that is output  
10 from DBS block 209 via lines 239. In the non-work conserving mode, if there is no  
11 traffic for the output port designated by the port calendar, then there will be no FID  
12 sent from DBS block 209 to PFQ block 207 during that sixteen clock cycle period.

13 A work-conserving mode is therefore provided. In the work-conserving mode,  
14 the port calendar checks the status of the next port in the port calendar to see  
15 whether traffic is waiting to be output from that next output port. A  
16 SCH\_AVAILABLE register is maintained in the DBS block. There is one bit in this  
17 register for each of the 64 output ports. After a dequeue, PFQ block 207 send an  
18 "empty" indication back to scheduler block 211 to indicate whether the last packet of  
19 the flow has now been sent. The scheduler block 211 knows whether this "empty"  
20 flow is the last flow for the designated output port. If the "empty" flow is the last flow  
21 for the designated output port, then scheduler block 211 updates the contents of the  
22 SCH\_AVAILABLE register to indicate that the scheduler has no traffic waiting for that  
23 output port. There is also a SHP\_AVAILABLE register maintained by DBS block  
24 209. The SHP\_AVAILABLE register indicates whether any of the per-port output  
25 FIFOs 303 for each output port has traffic waiting for that output port. There is also  
26 an SPIO\_FULL register that indicates a "backpressure busy" condition in which so  
27 much traffic has been sent out on the output port that the output port is full (for  
28 example, the receiving egress MS-SAR is being overloaded due to too much traffic  
29 being sent out of that output port on the ingress MS-SAR).

30 In the work conserving mode, the port calendar 230 looks ahead to check the  
31 appropriate bits in the SCH\_AVAILABLE register, and SHP\_AVAILABLE register and  
32 SPIO\_FULL register to determine if there is traffic waiting for, and whether traffic

1 should be sent out of, the output port to be designated by the port calendar next. If  
2 there is no traffic waiting or if no traffic should be sent, then the port calendar skips  
3 that output port on the next sixteen clock cycle dequeue phase and selects an  
4 subsequent output port that does have traffic waiting. The number of FIDs output  
5 from DBS block 209 per unit time is therefore increased.

#### 7 SCHEDULER IN MORE DETAIL:

8 **Figure 19** is a diagram that illustrates how the weighted round robin scheme  
9 of selecting a QOS is carried out. In order to implement the weighted round robin  
10 algorithm, two groups of QOSs are maintained per port. One is the "active" group  
11 and the other is the "waiting" group. In Figure 19, the three-bit value ACTIVE\_PTR  
12 identifies the current QOS to be serviced in the "active" group. The three-bit value  
13 PREV\_QOS identifies the previous QOS just serviced in the "active" group".

14 In the input phase, strict priority QOSs that are not "empty" are linked into the  
15 waiting group. Strict priority QOSs are never present in the active group.

16 Weighted round robin QOSs pass between the active group and the waiting  
17 group. If a new weighted round robin QOS is to be put into a group due to an input  
18 phase, then the new QOS is put into the waiting group after the current cycle is  
19 done. When a weighted round robin QOS is placed into the waiting group (either  
20 upon an input phase or when being moved from the active group to the waiting  
21 group), its weight count is set to its original weight. The original weight of a QOS is  
22 calculated based on two values, a weight parameter which is stored per QOS in the  
23 QOS par/descriptor memory 233, and a WEIGHT\_QUOTA value which is a  
24 programmable value that applies to all QOSs. The original weight of a QOS is the  
25 product of these two values.

26 When an output port is to be serviced, the "waiting" group is checked to  
27 determine if there are any strict priority QOSs that are not empty. This is done by  
28 reading the QW\_EMPTY field. There is one bit in this QW\_EMPTY field for each  
29 QOS to indicate whether the QOS in the waiting group is "empty" or not. If there are  
30 any strict priority QOS in the waiting group that are not empty, then these QOS are  
31 serviced first.



1           When all strict priority QOSs in the waiting group are empty, then non-empty  
2 QOSs can be selected in weighted round robin fashion from the active group. This  
3 is done by reading the QA\_EMPTY field. There is one bit in the QA\_EMPTY field for  
4 each QOS in the active group to indicate whether that QOS is empty or not. The  
5 Q\_WEIGHT\_MF value stored for the QOS (see Figure 13) is a count down weight  
6 value of the amount of weight that the current QOS has left. After the current  
7 weighted round robin QOS is serviced, this Q\_WEIGHT\_MF value is decremented  
8 by WEIGHT\_QUOTA. After the current weighted round robin QOS is serviced, the  
9 ACTIVE\_PTR value is switched so that it points to the next weighted round robin  
10 QOS in the active group. When the count down weight value for a weighted round  
11 robin QOS reaches zero, then its weight is said to be exhausted. When a weighted  
12 round robin QOS in the active group has exhausted its weight, then it is moved to  
13 the waiting group. If the active group ever becomes empty, then all the non-strict  
14 priority QOSs in the waiting group are moved to the active group. When a non-strict  
15 priority QOS is placed into the active group, its Q\_WEIGHT\_MF weight count down  
16 value is reset to be it's original weight.

17           Once the QOS is selected, the associated FID linked to the selected QOS is  
18 determined by reading the F\_RP pointer of the selected QOS. The FID pointed to  
19 by F\_RP is sent to DBS block 209 as the scheduled FID. Upon this FID being sent  
20 to DBS block 209, there are two possibilities. The first possibility is that the linked  
21 list of FIDs is rotated. If the current cell being scheduled out is the last cell (in case  
22 of ATM traffic, every cell sent out will be marked as EOP), then the scheduler block  
23 211 receives an EOP signal from DBS block 209. Also, if the current packet is the  
24 last packet linked for this FID, then scheduler block 211 receives an "empty"  
25 indication from DBS block 209. If an EOP signal is received but the FID is indicated  
26 as "not-empty", then scheduler block 211 rotates the FID linked list. This is done by  
27 moving the just serviced FID from the head of the FID linked list to the tail of the FID  
28 linked list. The head pointer is changed to point to the next FID in the list, and the  
29 tail pointer F\_WP is changed to point to the just serviced FID. The next FID in the  
30 list therefore becomes the head of the linked list.

1           The other possibility is that the just serviced FID is removed from the FID  
2 linked list. This is accomplished by changing the read pointer to point to the next  
3 FID in the list.

4           To prevent the interleaving of packets, the scheduler continues to service a  
5 QOS until an EOP is received for that QOS. This continued servicing occurs  
6 irrespective of priority.

#### 8 SHAPER IN MORE DETAIL:

9           Shaper block 210 performs either single-leaky bucket shaping or dual-leaky  
10 bucket algorithm on an FID, depending on which one of a possible 4K sets of  
11 shaping profiles is provisioned to be the shaping profile for the particular FID. Up to  
12 32K FIDs (or aggregated FIDs) can be shaped simultaneously. Which of the 4K  
13 shaping profiles is used to shape an FID is determined by the value RATE\_ID (see  
14 Figure 4) stored for the FID. Figure 5 is a diagram of a shaping profile for one FID.  
15 The shaping profile includes several user-configurable values including: a threshold  
16 value THR, a "sustained rate" Ks, and a "peak rate" Kp. The units of THR is shaping  
17 credits. The units for Ks and Kp are timing wheel time slots. The sustained rate and  
18 the peak rate are stored as floating point numbers, so the shaping profile (see Figure  
19 5) contains an exponent portion and a mantissa portion for each.

20           For each FID, shaper block 210 maintains a "SHP credit" value (shaping  
21 credit). When an FID is to be linked to a timing wheel, the "SHP credit" value of the  
22 FID is checked. If the "SHP credit" value is less than the provisioned THR value for  
23 the FID, then the FID is to be shaped at the "sustained rate" Ks. If, on the other  
24 hand, the "SHP credit" value is more than the provisioned THR value for the FID,  
25 then the FID is to be shaped at the "peak rate" Kp. Once shaper block 210 has  
26 started shaping at the "peak rate" Kp, shaper block 210 continues shaping at the  
27 "peak rate" until the "SHP credit" value decreases to zero, at which point shaping at  
28 the "sustained rate" resumes.

29           If the "peak rate" and the "sustained rate" for an FID are provisioned to be the  
30 same, then effectively there is one rate and "single leaky bucket" shaping is  
31 implemented. Single leaky bucket shaping can also be set by writing a "0" to the  
32 PEAK\_SUSTAIN bit for the FID in shaper internal FID#1 memory 228 (see Figure 7).

1           If the “peak rate” is higher than the “sustained rate” and the PEAK\_SUSTAIN  
2 bit is set to a “1”, then “dual leaky bucket” shaping is implemented.

3           In one embodiment, to provision the MS-SAR, a user supplies the following  
4 parameters to a driver program: a SCR value (sustained rate in cells/time units), a  
5 PCR (peak rate in cells/time units), a MBS (maximum burst size in cell units) and a  
6 CDVT (cell delay variation time). The driver program converts these values into the  
7 following values: the Ks value (number of timing wheel slots ahead to put the FID in  
8 a sustained rate), the Kp value (number of timing wheel slots ahead to put the FID in  
9 a peak rate), and the THR rate (a number of “SHP credits”). These values are then  
10 provisioned into MS-SAR 124 via CPU interface block 212.

11           Traffic shaper portion 227 includes a 19-bit time measurement counter. This  
12 counter is incremented once every eight cycles of the 200 MHz clock (the timing  
13 wheels also rotate once every eight cycles). When an FID is removed from the  
14 output FIFO of a timing wheel and is sent to the appropriate per-port output FIFO  
15 303 in DBS block 209, the count of the counter used as a CURRENT timestamp.  
16 This CURRENT timestamp is compared with the timestamp recorded the last time  
17 this FID was similarly sent to DBS block 209. This last time value is retrieved from  
18 the LAST\_TIME field in the shaper internal FID#1 memory 228 (see Figure 7). The  
19 difference between the CURRENT timestamp and the LAST\_TIME timestamp is the  
20 amount of time that elapsed between the sending of this FID to DBS block 209 this  
21 time and the last. This elapsed time value is divided by eight (because there are  
22 eight clock cycles per slot time), and the desired number of counter cycles (the  
23 sustained Ks value) is subtracted to obtain the “SHP\_credit” value. If the elapsed  
24 time is smaller than the desired Ks value, then “SHP\_credit” is negative. If the  
25 elapsed time is greater than the desired Ks value, then the “SHP\_credit” value is  
26 positive. The “SHP\_credit” value so calculated is then added to the prior  
27 accumulated “SHP\_credit” value stored for this FID in the shaper internal FID#1  
28 memory 228 (see Figure 7). The resulting accumulated value is then written back  
29 into the “SHP\_credit” field in shaper internal memory 228.

30           If the “SHP\_credit” accumulated value exceeds the stored value THR, then  
31 the peak Kp shaping rate value is used to determine which slot of the timing wheel to  
32 reattach the FID to. If the “SHP\_credit” value does not exceed the stored value

1 THR, then the sustained Ks shaping rate value is used to determine which slot of the  
2 timing wheel to reattach the FID to.

3 Assume for illustration purposes here that the sustained Ks shaping rate is to  
4 be used. The FID cannot necessarily be reattached to the timing wheel Ks number  
5 of slots ahead. It may have been the case that this FID is one of many FIDs that  
6 were all attached to the same slot of the timing wheel. All these FIDs would then  
7 have been dumped into the output FIFO of the shaping wheel at once. Because  
8 only one FID can be moved from a shaping wheel output FIFO to DBS block 209 at  
9 a time, some of the FIDs may have stayed in the shaping wheel output FIFO for  
10 multiple time slot periods. If after this wait the FID were then reattached Ks slots in  
11 the future, then FID would be attached too far in the future.

12 To compensate for the amount of time an FID may have remained in a  
13 shaping wheel output FIFO, a timestamp is taken when the FID is placed (i.e.,  
14 arrives) into the output FIFO. This timestamp value is the ARRIVAL\_TIME value  
15 stored in shaper internal FID#1 memory 228 (see Figure 7). The ARRIVAL\_TIME  
16 value is subtracted the desired K (Ks, for example) value, and the resulting number  
17 K is the number of slots ahead in the timing wheel where the FID is reattached.

#### 18 19 TUNNELING:

20 MS-SAR 124 can be provisioned such that multiple selected ones of the  
21 regular traffic-carrying flows (called "leaf" FIDs) are aggregated together into a  
22 logical entity called a "root" FID or a "tunnel" FID. All the aggregated "leaf" FIDs  
23 associated with a "tunnel" FID can then be shaped together by shaping the "tunnel"  
24 FID. DBS block 209 implements this tunneling mechanism such that no other  
25 functional blocks with the MS-SAR are tunneling-aware. Up to 256K flows can be  
26 merged and shaped into up to 32K aggregated flows.

27 To implement tunneling, DBS block 209 includes two internal memories: a  
28 tunnel memory 241, and a leaf memory 242. **Figure 20** is a diagram of tunnel  
29 memory 241. There is one set of fields such as those shown in Figure 20 for each  
30 FID. Accordingly, an incoming FID can be used to look up the associated  
31 TUNNEL\_VALID field in tunnel memory 241 to determine whether the incoming FID  
32 is a tunnel or not. **Figure 21** is a diagram of leaf memory 242. There is one set of

1 fields such as those shown in Figure 21 for each FID. Accordingly, an incoming FID  
2 can be used to look up the associated LEAF\_VALID in leaf memory 242 to  
3 determine whether the incoming FID is a leaf FID or not.

4 **Figure 22** is a diagram of a linked list structure used to implement a tunnel  
5 FID. In the illustrated example, there are three leaf FIDs (FID1, FID2 and FID3)  
6 aggregated together into one tunnel FID (FID 4). The TUNNEL\_VALID field in  
7 tunnel memory 241 (see Figure 20) for the tunnel FID (FID 4) is set to indicate that  
8 FID 4 is a tunnel FID. The LEAF\_RP read pointer points to the first leaf FID (FID 1  
9 in this example) of the linked list of leaf FIDs of this tunnel. The LEAF\_WP write  
10 pointer points to the last leaf FID (FID 3 in this example) of the linked list of leaf FIDs  
11 of this tunnel. A leaf FID is made to point to the next leaf FID in the list by writing to  
12 the NEXT\_LEAF field in the leaf memory of the leaf FID. In the present example,  
13 the NEXT\_LEAF field in leaf memory 242 for FID 1 is made to point to FID 2.

14 To illustrate operation of tunneling, an example of an input phase is described  
15 wherein an FID is passed from CBWFQ block 208 to DBS block 209. If the incoming  
16 FID is a leaf of a tunnel and was empty before, then DBS block 209 links the FID to  
17 the appropriate tunnel linked list and sends the tunnel FID out of DBS block 209 to  
18 shaper block 210 in accordance with the input phase set forth above. DBS block  
19 209 determines whether the incoming FID is leaf and whether the FID is empty by  
20 examining the LEAF\_VALID and LEAF\_EMPTY fields, respectively, for the incoming  
21 FID in leaf memory 241. If the incoming FID is determined to be a leaf, DBS block  
22 209 identifies the tunnel FID for the leaf by reading the TUNNEL\_PTR field in leaf  
23 memory 241. This field stores a pointer to the tunnel FID for this leaf FID.

24 Tunnel FIDs are not scheduled. Consequently, if a tunnel FID having leaves  
25 is to be output from DBS block 209, then DBS 209 sets the two bits accompanying  
26 the tunnel FID to indicate that the FID forwarded is to be received for an input phase  
27 by shaper block 210 but not by scheduler block 211. Shaper block 210 receives the  
28 FID from DBS block 209 and shapes the FID as if it were a regular FID having no  
29 leaves.

30 In the case where the forwarded FID is a tunnel with leaves, and shaper block  
31 210 shapes the tunnel, the tunnel is then forwarded to the per-port output FIFOs 303  
32 of DBS block 209 as described above. On an output phase of DBS block 209, when

1 the FID is selected out of the per-port output FIFO, DBS block 209 checks tunnel  
2 memory 241. If the FID is not a tunnel, then the FID is forwarded to PFQ block 207  
3 via CBWFQ block 208.

4 If, on the other hand, the FID is a tunnel with leaves as determined by the  
5 contents of the tunnel memory, then DBS block 209 looks up the first leaf FID in the  
6 linked list of leaves (the leaf pointed to by LEAF\_RP) and sends that FID out to PFQ  
7 block 207 via CBWFQ block 208. If an EOP is received from PFQ block 207, then  
8 DBS block 209 moves the leaf that was sent out from the head of the linked list to  
9 the tail of the linked list (i.e., rotates the linked list) by changing the LEAF\_RP pointer  
10 to point to the next leaf in the list, by changing the last leaf in the list to point to the  
11 leaf that was sent out, and by changing the LEAF\_WP to point to the leaf that was  
12 sent out. Accordingly, for a given tunnel FID received from shaper block 210, leaf  
13 FIDs are selected for passing to CBWFQ block 208 in round robin fashion.

14 If tunnel FIDs were to be allocated from the normal FID space, then a loss of  
15 FIDs would result. The number of FIDs available for use as regular unicast FIDs or  
16 another leaf FID would be reduced. To avoid this problem, the tunnel FID can be  
17 chosen as one of the leaf FIDs. This way, whenever a set of leafs are being  
18 tunneled, FID space does not have to be wasted to allocate a tunnel FID. Rather,  
19 the tunnel FID is selected as one of the leafs. Because FIDs can be shared  
20 between tunnels and leafs, however, care is taken to interpret FIDs correctly. Only  
21 leaf FIDs are exchanged between DBS block 209 and CBWFQ block 208. Only  
22 tunnel FIDs (with leaves or without leaves) can be exchanged between DBS block  
23 209 and shaper block 210. It is an invalid condition to receive a tunnel FID from the  
24 PFQ. It is an invalid condition to receive a leaf FID from the shaper.

#### 25 26 CBWFQ BLOCK:

27 CBWFQ (Class-Based Fair Weighted Queueing) merges a number of flows  
28 into one root. The flows that are serviced are called CBWFQ leaf flows and the  
29 aggregate is called the CBWFQ root flow or virtual circuit (VC). The root flow is a  
30 regular flow which can be shaped (with or without funneling) or scheduled just like  
31 any other flow. The CBWFQ feature is typically used when multiple flows are to be  
32 merged onto one single ATM VC.

1           As in the case of tunneling described above, aggregated flows are stored in  
2 the form of linked lists of FIDs. When a merged flow is scheduled to be dequeued  
3 by the scheduling algorithms, one of the leafs is selected to be dequeued based on  
4 one of four algorithms: 1) round robin (RR), 2) deficit round robin (DRR), 3) Alternate  
5 modified deficit round robin (MDRR), and 5) strict priority and modified deficit round  
6 robin.

7           CBWFQ block 208 utilizes two memories: external CBWFQ leaf descriptor  
8 memory 217, and an internal root (VC) descriptor memory 243. **Figure 23** is a  
9 diagram of external leaf CBWFQ descriptor memory 217. **Figure 24** is a diagram of  
10 internal VC (root) descriptor memory 243. **Figure 25** is a diagram that shows how  
11 the merged FIDs of a VC are maintained in a linked list form.

12           In an input phase, an FID is received from PFQ block 207. If the incoming  
13 FID is a leaf, and if the leaf is empty (there is not traffic pending from this leaf FID),  
14 then CBWFQ block 208 marks the leaf as "not empty", looks up the associated root,  
15 links the incoming FID into the linked list of the root, and then marks the root as "not  
16 empty". Designating the root as "not empty" means that there is a linked list of leafs  
17 (non empty leaves) for the root. CBWFQ block 208 then sends the root FID to DBS  
18 block 209. This entire operation is bypassed if the FID does not belong to a root  
19 FID.

20           In an output phase, CBWFQ block 208 receives an FID from DBS block 209.  
21 If the FID is a root FID, the CBWFQ selects one of the leaf FIDs to be sent to PFQ  
22 block 207. If in response to sending a leaf FID to PFQ block 207 an empty  
23 indication is received back, then CBWFQ block 208 remove the leaf FID from the  
24 linked list of FIDs for its root. If an EOP indication is received from PFQ block 207,  
25 then CBWFQ block 208 rotates the linked list of FIDs in accordance with the  
26 particular algorithm selected. The rotation is performed in similar fashion to the way  
27 the linked list of Figure 22 was rotated. The entire operation of CBWFQ block 208  
28 is bypassed if the FID received from DBS block 209 is not a root FID (VC).

29           RR: This is a simple round robin scheme. Once an EOP indication arrives  
30 from the PFQ block 207, the linked list of leaf FIDs is rotated.

31           DRR algorithm: This is a weighted round robin algorithm with the ability to  
32 support negative credit. Once an EOP indication arrives from PFQ block 207, if the

1 FID has a zero or negative weight it will be rotated to the end of the linked list.  
2 When this FID comes up for servicing again, if credit is still negative, then no output  
3 phase is performed but rather a new weight quota is added and is pushed back to  
4 end of link.

5 MDRR algorithm: This is an extension of the DRR algorithm. One FID is  
6 considered to be of higher priority than the others. It is therefore not linked to the  
7 list. The rest of the FIDs are considered as one group. There is a pure round robin  
8 between this high priority FID and the group so that the scheduling look like: FID,  
9 group, FID, group, FID, group, and so forth. When it is the turn of the group, an FID  
10 is selected based on the DRR algorithm.

11 Priority and DRR and Discard: This is another extension to DRR. This mode  
12 is the same as the previous one, except that if the high priority FID is not empty,  
13 then it is sent to PFQ block 207 without consideration to its weight. Only if the high  
14 priority FID is empty will the rest of the FIDs be transferred to the PFQ block 207  
15 based on the DRR scheme.

16

#### 17 EXAMPLE OF TRAFFIC MANAGEMENT CAPABILITIES:

18 **Figure 26** illustrates an example of some of the traffic management  
19 capabilities of MS-SAR 124 wherein an FID is selected and is supplied to PFQ block  
20 207 in an output phase. Portion 306 is generally considered to be a shaping  
21 function whereas portion 307 is generally considered to be a scheduling function.  
22 Bubble 308 represents the operation of port calendar 230. As set forth in the  
23 description of the port calendar above, the output phase starts with port calendar  
24 230 selecting an output port. Which output ports are selected and in what order is  
25 determined by how port calendar 230 is provisioned. Port calendar 230 in the  
26 example of Figure 26, selects one of the output ports represented in the diagram as  
27 lines extending from the left of bubble 308. In the example of Figure 26, the top  
28 output port (port number 0) is selected.

29 Once port 0 is selected, the selection proceeds to the left to bubble 309.  
30 Bubble 309 represents the selection by DBS block 209 of an FID from one of the  
31 per-port output FIFOs from one of the eight shaper timing wheels (represented here  
32 by the eight lines numbered 0-7 that extend to the left from bubble 309), or if there is



1 no FID output by the shaper block then an FID output by scheduler block 211 is  
2 selected (represented here by the bottom line numbered 7 that extends downward  
3 and to the left from bubble 309). Priorities 0-7 are for shaped traffic. The selection  
4 of FIDs from the per-port FIFOs of wheels 0 through 7 are by strict priority. This is  
5 represented by arrow 310. Priority 8 is for scheduled traffic.

6 Portion 311 represents shaping done by shaper wheel 0 (the highest priority  
7 shaping wheel). The "RR" in bubble 312 represents the round robin algorithm, and  
8 the bucket symbol 313 represents leaky bucket shaping (either single leaky bucket  
9 or dual leaky bucket). A shaping wheel can be provisioned to shape three types of  
10 elements: 1) ordinary FIDs, 2) tunnel root FIDs, and 3) MDRR root FIDs.

11 In the particular example of Figure 26, if shaper 0 selects an FID on line 315  
12 then a tunnel FID is shaped. As set forth in the description of tunneling above, when  
13 the tunnel FID passes through DBS block 209, one of its leaf FIDs is selected, and  
14 the selected leaf FID is then output from DBS block 209 to CBWFQ block 208. In  
15 the example of Figure 26, tunnel symbol 316 has three associated leaf FIDs. These  
16 leaf FIDs are represented in Figure 26 by the three lines extending to the left from  
17 tunnel symbol 316.

18 A tunnel can be set up to aggregate regular FIDs and MDRR elements.  
19 Tunnel 316 in Figure 26 illustrates this. Tunnel 316 aggregates two MDRR elements  
20 317, 332 and one regular FID 333. If the upper leaf FID is selected by the tunnel  
21 mechanism, the resulting FID in the example of Figure 26 is actually an "MDRR"  
22 FID. As set forth above, CBWFQ block 208 receives an MDRR root FID and selects  
23 one of the associated leaf FIDs. In the example of Figure 26, MDRR 317 has three  
24 associated leaf FIDs. Which of these leaf FIDs is selected depends on how the  
25 MDRR root flow is provisioned.

26 In addition to selecting a tunnel FID, a shaper wheel can also shape an  
27 ordinary FID. This is illustrated in Figure 26 by FID 318. A shaper wheel can also  
28 shape an MDRR root FID. This is illustrated in Figure 26 by MDRR 319.

29 Once DBS block 209 receives an EOP, DBS block 209 can select an FID  
30 from the highest priority per-port output FIFO of shaper block 210. In the example of  
31 Figure 26, if there is no such FID in the per-port output FIFO for shaper wheel 0,  
32 then an FID can be taken from the per-port output FIFO for shaper wheel 1

1 (represented by the line labeled "1" extending to the left from priority bubble 309).  
2 Similarly, if there is no FID in any of the per-port output FIFOs for shaper wheels 0-5,  
3 then DBS block 209 can select an FID from shaper wheel 7. Shaper wheel 7 is  
4 represented by portion 320.

5 If there is no FID to select from shaper block 210, then an FID can be  
6 supplied via line 321 from scheduler block 211. The lines extending from the left of  
7 priority/DRR bubble 322 represent the QOS classes that may be provisioned. As set  
8 forth in the description of scheduler block 211 above, a number of the highest priority  
9 QOSs can be provisioned to be selected between using a strict priority scheme, and  
10 the remaining QOSs (but for QOS 7) can be provisioned to be selected between  
11 using a weighted round robin scheme. QOS 7 is selected on a best efforts basis.  
12 For the QOS selected, the scheduler selects an element from a linked list linked to  
13 the selected QOS. Two types of elements can be scheduled: 1) regular FIDs, and 2)  
14 MDRR root FIDs. Which element is selected is determined using a round robin  
15 scheme. This is represented in Figure 26 by the "RR" in the bubbles 323 and 324 to  
16 the left of the QOS numbers. An element in one of these linked lists can be an  
17 MDRR root. This is illustrated in Figure 26 by line 325 extending to the left from  
18 bubble 323 to MDRR symbol 326. When the MDRR root FID 326 passes from  
19 scheduler 211 through CBWFQ block 208, CBWFQ block 208 selects one of the leaf  
20 FIDs associated with MDRR root FID 326. These leaf FIDs are represented in  
21 Figure 26 by the three lines extending to the left from MDRR symbol 326. CBWFQ  
22 block 208 then selects one of these leaf FIDs in accordance with the algorithm  
23 provisioned for the root, and forwards that selected leaf FID to PFQ block 207.

24 MS-SAR 124 can be provisioned to both shape and schedule an FID. This is  
25 represented by FID 327 passing to the right via line 328 to shaper wheel 0 or  
26 passing down to QOS 7 being scheduled via line 329. Note that this FID 327 that is  
27 both shaped and scheduled may be an MDRR flow as indicated by MDRR symbol  
28 330.

29 The FID produced by the traffic management structure of Figure 26 for the  
30 selected output port is then supplied to PFQ block 207 for dequeuing. This is  
31 represented in Figure 26 by arrow 331.

1           Although the present invention is described in connection with certain specific  
2   embodiments for instructional purposes, the present invention is not limited thereto.  
3   Accordingly, various modifications, adaptations, and combinations of various  
4   features of the described embodiments can be practiced without departing from the  
5   scope of the invention as set forth in the claims.